UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/782,715	02/19/2004	Kevin Zatloukal	BEAS-01396US1 SRM/DTX	5577

23910 7590 11/10/2004

FLIESLER MEYER, LLP
FOUR EMBARCADERO CENTER
SUITE 400
SAN FRANCISCO, CA 94111

EXAMINER	
CHOW, CHIH CHING	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 11/10/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/782,715	ZATLOUKAL, KEVIN	
	Examiner	Art Unit	
	Chih-Ching Chow	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 February 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-56* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *19 February 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed on February 19, 2004.

2. The priority date considered for this application is February 26, 2003, which is the filing date of the provisional application no. 60/449,991.

3. Claims 1-56 have been examined.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless -
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1, 7-8, 19, 25-26, 37, 43-44, and 55-56 are rejected under 35 U.S.C. 102(b) as being anticipated by Robert Neil Faiman Jr., US Patent No. 5, 836,014 (hereinafter "Faiman").

CLAIM	Faiman
1. A system operable to perform a multi-programming-language compilation process on a computer program, comprising:	Faiman teaches a compiler to perform a multi-programming-language compilation. In Faiman's Fig. 1, shows an operable **compiler system**; in Faiman's Abstract, "A **compiler framework** comprises a generic compiler back end which may be used by a **plurality of front ends to generate object code for a target computer system. Each front end** scans and parses a source module containing source code **for a programming language**". For items (a) and (b), in Faiman, column 3, lines 45-49, "Various higher level **languages** have differing ways of expressing operations, and the same sequence may in one language allow a result or dependency, while in another language it may not. Thus, a mechanism which **is independent of source language** (*programming language independent portion*) is provided for describing the effects of program execution. This mechanism provides **a means for the compiler front end** to generate a detailed **language-specific information** (*programming language-dependent portion*) to the **multi-language** optimizer in the compiler back end"; further, in column 3, lines 59-61, "The interface between back end and front end is in this respect language **independent**. The back end does not need to know what language it is compiling." Lines, 63-65, "back end **need not be written for each source**
(a) a compiler framework operable to perform a programming language independent portion of the compilation process;	
(b) a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process; and	

	language, but instead an optimizing compiler can be produced for each source language by merely tailoring a front end for each different language."
(c) a plurality of language interfaces, wherein each language interface is provided by a language module to interact with the compiler framework.	For item (c), in Faiman's Abstract, "A **compiler framework** comprises a generic compiler back end which may be used by a **plurality of front ends** (*a plurality of language modules, a plurality of language interfaces*) to generate object code for a target computer system." Faiman teaches all aspects of claim 1.
7. The system of claim 1, wherein: the programming language-dependent portion of the compilation process comprises at least one of the following phases; (a) performing lexical analysis; (b) performing syntactic analysis; (c) performing name resolution; (d) performing semantic analysis; and (e) performing code generation.	For the feature of claim 1 see claim 1 rejection. In Faiman, column 6, lines 24-26, "the front end 20 (*language-dependent portion*) does **lexical**, **syntactic**, and **semantic analysis** to translate the source text in file 21 to a language-independent internal representation used for the interface 22 between the front end 20 and the back end 12.", column 6, lines 65-67, "The back end 12 performs the basic functions of optimization 26, **code generation** 27, storage and register allocation 28, and object file emission 29." In Faiman's disclosure, the programming-dependent portion (front end) does 'lexical, syntactic, and semantic analysis' of the compilation process.
8. The system of claim 7, wherein: a language interface of the plurality of language interfaces may present	Same as claim 7 rejection.

language-dependent portion of the compilation process in the form of a set of components, each component performing one phase of the compilation process.	
19. A method operable to perform a multi-programming-language compilation process on a computer program, comprising: (a) utilizing a compiler framework operable to perform a programming language-independent portion of the compilation process; (b) invoking a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process; and (c) providing a plurality of language interfaces, wherein each language interface is operable to allow a language module to interact with the compiler framework.	Same as claim 1 rejection.
25. The method of claim 19, wherein: the programming language-dependent portion of the compilation process comprises at least one of the following phases; (a) performing lexical analysis; (b) performing syntactic analysis; (c) performing name resolution; (d) performing semantic analysis; and (e) performing code generation.	For the feature of claim 19 see claim 19 rejection. For the rest of the features of claim 25, see claim 7 rejection.

26. The method of claim 25, further comprising: presenting the language-dependent portion of the compilation process in the form of a set of components, each component performing one phase of the compilation process.	For the feature of claim 25 see claim 25 rejection. For the rest of the features of claim 26, see claim 8 rejection.
37. A machine readable medium having instructions stored thereon that when executed by a processor cause a System to: (a) perform a multi-programming-language compilation process on a computer program, comprising; (b) utilizing a compiler framework operable to perform a programming language-independent portion of the compilation process; (c) invoking a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process; and (d) providing a plurality of language interfaces, wherein each language interface is operable to allow a language module to interact with the compiler framework.	In Faiman's Fig. 2, items 15 and 17 show a **machine readable medium**. For the rest of the features in claim 37, same as claim 1 rejection.
43. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: perform the programming language-dependent portion of the compilation	For the feature of claim 37 see claim 37 rejection. For the rest of the features of claim 43, see claim 7 rejection.

process in at least one of the following phases;
(a) performing lexical analysis;
(b) performing syntactic analysis;
(c) performing name resolution;
(d) performing semantic analysis; and
(e) performing code generation.

44. The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:
present the language-dependent portion of the compilation process in the form of a set of components, each component performing one phase of the compilation process.

For the feature of claim 43 see claim 43 rejection. For the rest of the features of claim 44, see claim 8 rejection.

55. A system operable to perform a multi-programming-language compilation process on a computer program, comprising:
(a) means to utilize a compiler framework operable to perform a programming language-independent portion of the compilation process;
(b) means to invoke a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process; and
(c) means to provide a plurality of language interfaces, wherein each language interface is operable to allow a language module to interact with the compiler framework.

Same as claim 1 rejection.

56. A computer data signal embodied in a transmission medium, comprising:
(a) a code segment including instructions to perform a multi-programming-language compilation process on a computer program, comprising;
(i) utilizing a compiler framework operable to perform a programming language-independent portion of the compilation process;
(ii) invoking a plurality of language modules, wherein each language module is operable to perform a programming language-dependent portion of the compilation process; and
(iii) providing a plurality of language interfaces, wherein each language interface is operable to allow a language module to interact with the compiler framework.

See for example the **signal transmitted** between items 23 and 25 of Famain's Fig. 1. For the rest of the features of claim 56, same as claim 1 rejection.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 2-4, 6, 9, 14-18, 20-22, 24, 27, 32-36, 38-40, 42, 45, and 50-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 5, 836,014 by Robert Neil Faiman Jr. (hereinafter "Faiman"), in view of U.S. Patent No. 6,804,686 by Blake W. Stone et al. (hereinafter "Stone").

CLAIM	Faiman / Stone
2. The system of claim 1, wherein: a multi-threading service is used by the compiler framework and the plurality of language modules. 3. The system of claim 1, wherein: the system is tailored for the Java environment.	For the feature of claim 1 see claim 1 rejection. Faiman teaches all aspects of claim 2 and 3, but does not mention the 'multi-threading' (claim 2) and 'Java environment' (claim 3) specifically. However, Stone teaches these features in an analogous art. In Stone, column 8, lines 12-24, "**Java** is a simple, object-oriented language which supports **multi-thread** processing and garbage collection. ...**Java programs** are 'compiled' into a binary format that can be executed on many different platforms without recompilation. A typical **Java** system comprises the following set of interrelated technologies: a language specification; a compiler for the **Java** language that produces bytecodes from an abstract, stack-oriented machine; a virtual machine (VM) program that interprets the bytecodes at runtime; a set of class libraries; a runtime environment that includes bytecode verification, **multi-threading**, and garbage collection;" It would have been obvious to a person of ordinary skill in the art at the time of

	the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by utilizing Java environment and multi-threading service taught by Stone, for the purpose of deploying application code efficiently to multiple platforms. (Stone, column 2, lines 7-8).
4. The system of claim 1, wherein: the computer program is organized into a project, which may contain at least one set of the following: files, paths, libraries, configuration information, and dependencies among files.	For the feature of claim 1 see claim 1 rejection. Faiman teaches all aspects of claim 4 but does not mention the 'organizing program into a project' specifically. However, Stone teaches these features in an analogous art. In Stone, column 9, lines 40-42, "To develop a software program in the development environment, a user typically first creates a '**project**' to organize the **program files** and maintain the **properties set** (*files, paths, configuration information*) for the program". In column 8, lines 42-44, "As shown in FIG. 3B, the virtual machine 320 comprises a class loader 321, a bytecode verifier 322, a bytecode interpreter 323, and runtime support **libraries** 324."; further more, in column 9, lines 1-10, "Runtime support **libraries** 324 comprise **functions** (typically, written in C) which provide runtime support to the virtual machine, including memory management, synchronization, type checking, and interface invocation." In column 10, lines 8-10, "The content pane 481 provides access to various **file**

views as well as **status information** (*configuration information*) by way of file view tabs 485 and a file status bar 486". In column 11, lines 36-38, "The repository 520 contains information regarding **relations between components** (*dependencies*) of a software program or application. These relations include superclasses, interfaces, associations, ***dependencies*** (both incoming and reverse), and subclasses.", column 12, lines 36-37, "A '**dependency**' is a using **relationship** in which a change to an object may affect another dependent object.", column 13, lines 56-57, "**Dependency**" Java definition, and column 14, lines 59- 67, 'Package **Dependency** Diagram'.
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by utilizing the 'project' concept taught by Stone, for the purpose of organizing program files and maintaining the properties set for the program. (see Stone, column 9, lines 42-43).

6. The system of claim 4, wherein:
the programming language-independent portion of the compilation process comprises at least one of the following phases:
(a) managing the set of files in the project;

For the feature of claim 4 see claim 4 rejection. Faiman teaches all aspects of claim 6 but does not mention ' a list of errors' specifically. However, Stone teaches these features in an analogous art. For item (a)-(e) see claim rejection 4; for item (f), in Stone column 9, lines

(c) persisting the set of paths files in the project;
(d) maintaining the set of dependencies in the project;
(e) acquiring configuration information files in the project; and
(f) maintaining a list of errors related to the project.

58-59, "When appropriate, the structure pane 475 also displays an **'Errors' folder** (*list*) containing any syntax errors in the file".
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by displaying the error taught by Stone, for the purpose of acknowledging the users the properties of the program. (see Stone, column 9, lines 42-43).

9. The system of claim 1, wherein:
a language interface of the plurality of language interfaces may include functions for retrieving information about a particular file in the computer program.

For the feature of claim 1 see claim 1 rejection. Faiman teaches all aspects of claim 9 but does not mention the 'retrieving information' specifically. However, Stone teaches these features in an analogous art. In Stone, column 9, lines 65-67, "The **content pane** 481 (*language interface*) **displays all open files** in a project as a set of tabs. Files may be **opened** (*functions for retrieving information*) in the content pane 481 by selecting the file from the project pane 471".
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by retrieving information about a file taught by Stone, for the purpose of providing access to source file to enable modifications via the editor (see Stone,

column 10, lines 5-7).

14. The system of claim 1, further comprising:
a tool to speed the development of the plurality of language modules.

For the feature of claim 1 see claim 1 rejection. Faiman teaches all aspects of claim 14 but does not mention the 'a tool to speed the development' specifically. However, Stone teaches these features in an analogous art. In Stone, column 10, lines 3-6, "A user may select a file tab to display a particular file in the content pane 481. The content pane provides a **full-featured editor** that provides **access to text** (i.e., source code) in a given project." (*first module interact with second module*). - an editor is definitely a **tool** for speed the development of program module.
Also in Stone column 9, lines 49-53, "In the currently preferred embodiment, the project pane 471 also includes a project pane **toolbar** 472 which includes buttons for closing a project, adding files or packages (e.g., by opening an 'Add Files or Packages to Project' dialog box), removing files from a project, and refreshing the project" (*tool to speed the development*). Further, in column 13, lines 29-31, "UML, a **valuable tool** in understanding complex relationships between elements of a system".
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by using tool to speed up the development by

	Stone for the purpose of assisting developers in understanding the overall design and structure for a software system (see Stone, column 13, lines 32-33).
15. A system operable to perform a multi-programming-language compilation process on a computer program, comprising: (a) a compiler framework operable to perform a language-independent portion of the compilation process; (b) a plurality of language modules, each language module in the plurality of language modules is operable to perform a language-dependent portion of a compilation process and to provide language information about the computer program; (c) an information interface operable to permit each client in a plurality of clients to interact with the compiler framework; (d) a plurality of clients operable to utilize the information interface to request the compiler framework to perform at least one of the following tasks; obtaining the language information produced by the plurality of language modules; and requesting a service provided by the compiler framework.	For items (a)-(c) same as claim 1 rejection. For item (d), Stone's content pane provides 'language information' and services, see claim 4 and claim 9 rejections.

16. The system of claim 15, wherein the plurality of clients comprise:
an integrated development environment (IDE) for developing a computer program wherein the IDE uses the language information from the compiler framework to provide language features based on that information; and

a command-line interface for causing the compilation of the computer program and displaying any errors in the computer program.

For the feature of claim 15 see claim 15 rejection. Faiman teaches all aspects of claim 16 but does not mention the 'independent environment (IDE)' specifically. However, Stone teaches IDE in an analogous art. In Stone, column 2, lines 36-40, "Such environments are characterized by an **integrated development environment (IDE)** providing a form painter, a property getter/setter manager ("inspector"), a project manager, a **tool** palette (with objects which the user can drag and drop on forms), an **editor**, and a **compiler**". In column 7, lines 59-60, "System 200 includes a graphical user interface (GUI) 215, for **receiving user commands**". For 'displaying errors' see claim 6 (f) rejection.
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by using IDE taught by Stone for the purpose of providing a visual development environment to simplify the development process by quickly creating production applications (see Stone, column 2, lines 33-37).

17. The system of claim 16, wherein:
the IDE may include a source code editor to edit files in the computer program.

Same as claim 16 rejection.

18. The system of claim 15, wherein:

For the feature of claim 15 see claim 15

the information interface is further operable to allow a client to inform the compiler network of file changes; and the compiler network is operable to recompile the changed files and other files depend on them.	rejection. Faiman teaches all aspects of claim 18 but does not mention the 'recompiling the changed files' specifically. However, Stone teaches IDE in an analogous art. In Stone, column 18, lines 38-40, "if a source file has been changed but has not been **recompiled**, a message is displayed in the UML browser indicating that the UML diagram may not be accurate." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman disclosure of the multi-programming-language compiler by recompiling changed files by Stone for the purpose ensuring the program is up-to-date (see Stone, column 18, lines 28-30).
20. The method of claim 19, further comprising: utilizing a multi-threading service during the compilation process.	For the feature of claim 19 see claim 19 rejection. For 'multi-threading' feature see claim 2 rejection.
21. The method of claim 19, further comprising: tailoring the compilation process for the Java environment.	For the feature of claim 19 see claim 19 rejection. For 'Java' feature see claim 3 rejection.
22. The method of claim 19, wherein: the computer program is organized into a project, which may contain at least one set of the following: files, paths, libraries, configuration information, and dependencies among files.	For the feature of claim 19 see claim 19 rejection. For 'project' features see claim 4 rejection.

24. The method of claim 22, wherein: the programming language independent portion of the compilation process comprises at least one of the following phases; (a) managing the set of files in the project; (b) persisting the set of paths files in the project; (c) maintaining the set of dependencies in the project; (d) acquiring configuration information files in the project; and (e) maintaining a list of errors related to the project.	For the feature of claim 22 see claim 22 rejection. For the rest of the features see claim 6 rejection.
27. The method of claim 19, further comprising: retrieving information about a particular file in the computer program via a language interface of the plurality of language interfaces.	For the feature of claim 19 see claim 19 rejection. For rest of the features see claim 9 rejection.
32. The method of claim 19, further comprising: adopting tools to speed up the development of the plurality of language modules.	For the feature of claim 19 see claim 19 rejection. For rest of the features see claim 14 rejection.
33. A method operable to perform a multi-programming-language compilation process on a computer program, comprising: (a) utilizing a compiler framework operable to perform a language	For items (a)-(c) same as claim 1 rejection. For item (d), Stone's content pane provides 'language information' and services, see claim 4 and claim 9 rejections.

independent portion of the compilation process;

(b) invoking a plurality of language modules, each language module in the plurality of language modules is operable to perform a language-dependent portion of a compilation process and to provide language information about the computer program;

(c) providing an information interface operable to permit each client in a plurality of clients to interact with the compiler framework;

(d) including a plurality of clients operable to utilize the information interface to request the compiler framework to perform at least one of the following tasks;

(i) obtaining the language information produced by the plurality of language modules; and

(ii) requesting a service provided by the compiler framework.

34. The method of claim 33, wherein the plurality of clients comprise:

(a) an integrated development environment (IDE) for developing a computer program wherein the IDE uses the language information from the compiler framework to provide language features based on that information; and

(b) a command-line interface for causing the compilation of the computer program and displaying any errors in the computer program.

For the feature of claim 33 see claim 33 rejection. For rest of the features see claim 16 rejection.

35. The method of claim 34, further comprising: including a source code editor in the IDE to edit files in the computer program.	For the feature of claim 34 see claim 34 rejection. For rest of the features see claim 17 rejection.
36. The method of claim 33, further comprising: allowing a client to inform the compiler network of file changes; and recompiling the changed files and other files depend on them.	For the feature of claim 33 see claim 33 rejection. For rest of the features see claim 18 rejection.
38. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: use a multi-threading service during the compilation process.	For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 2 rejection.
39. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: tailor the compilation process for the Java environment.	For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 3 rejection.
40. The machine readable medium of claim 37, wherein: the computer program is organized into a project, which may contain at least one set of the following: files, paths, libraries, configuration information, and dependencies among files.	For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 4 rejection.

42. The machine readable medium of claim 40, further comprising instructions that when executed cause the system to: perform the programming language-independent portion of the compilation process in at least one of the following phases; (a) managing the set of files in the project; (b) persisting the set of paths files in the project; (c) maintaining the set of dependencies in the project; (d) acquiring configuration information files in the project; and (e) maintaining a list of errors related to the project.	For the feature of claim 40 see claim 40 rejection. For rest of the features see claim 6 rejection.
45. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: retrieve information about a particular file in the computer program via a language interface of the plurality of language interfaces.	For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 9 rejection.
50. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to: adopt tools to speed up the development of the plurality of language modules.	For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 14 rejection.
51. The machine readable medium of	For the feature of claim 37 see claim 37

claim 37, further comprising instructions that when executed cause the system to:

(a) utilize a compiler framework operable to perform a language-independent-portion of the compilation process;

(b) invoke a plurality of language modules, each language module in the plurality of language modules is operable to perform a language-dependent portion of a compilation process and to provide language information about the computer program;

(c) provide an information interface operable to permit each client in a plurality of clients to interact with the compiler framework;

(d) include a plurality of clients operable to utilize the information interface to request the compiler framework to perform at least one of the following tasks;

(i) obtaining the language information produced by the plurality of language modules; and

(ii) requesting a service provided by the compiler framework.

rejection. For items (a)-(c) same as claim 1 rejection. For item (d), Stone's **content pane** provides '**language information' and services**, see claim 4, 15, and claim 9 rejections.

52. The machine readable medium of claim 37, wherein the plurality of clients comprise:

(a) an integrated development environment (IDE) for developing a computer program wherein the DE uses the language information from the

For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 16 rejection.

compiler framework to provide language features based on that information; and (b) a command-line interface for causing the compilation of the computer program and displaying any errors in the computer program.	
53. The machine readable medium of claim 52, further comprising instructions that when executed cause the system to: include a source code editor in the IDE to edit files in the computer program.	For the feature of claim 52 see claim 52 rejection. For rest of the features see claim 17 rejection.
54. The machine readable medium of claim 51, further comprising instructions that when executed cause the system to: allow a client to inform the compiler network of file changes; and recompile the changed files and other files depend on them.	For the feature of claim 52 see claim 52 rejection. For rest of the features see claim 18 rejection.

7. Claims 5, 23 and 41 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 5, 836,014 by Robert Neil Faiman Jr. (hereinafter "Faiman"), in view of U.S. Patent No. 6,804,686 by Blake W. Stone et al. (hereinafter "Stone"), further in view of U.S. Patent No. 6,732,237 by Lawrence Jacobs et al. (hereinafter "Jacobs").

CLAIM	Faiman / Stone / Jacobs
5. The system of claim 4, further comprising a type cache operable to: store types defined in the set of files in the project; store dependencies between the types it stores; and allow types defined in one programming language to reference types defined in another programming language.	For the feature of claim 4 see claim 4 rejection. Stone teaches all aspects of claim 5, but he does not mention 'type cache' specifically, however, Jacobs teaches it in an analogous prior art. In Jacobs' column 2, lines 48-51, "An analysis engine may operate under a set of guidelines or rules to determine, what data should be **stored** in which **type of cache**, and/or route data requests to one cache or the other', further, in column 14, lines 30-33, "the administrator may specify that data requests having a specified form may need to be served in a number of **different languages** depending on a particular value provided with the requests." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman and Stone's disclosure of the multi-programming-language Java environment by using type cache storage taught by Jacobs, for the purpose of performance enhancement for data requesting (Jacobs column 2, lines 14-30).
23. The method of claim 22, further comprising: (a) utilizing a type cache operable to; (b) store types defined in the set of files in the project; (c) store dependencies between the	For the feature of claim 22 see claim 22 rejection. For the rest of the features see claim 5 rejection.

types it stores; and
(d) allow types defined in one programming language to reference types defined in another programming language.

41. The machine readable medium of claim 40, further comprising instructions that when executed cause the system to:
(a) utilize a type cache operable to;
(b) store types defined in the set of files in the project;
(c) store dependencies between the types it stores; and
(d) allow types defined in one programming language to reference types defined in another programming language.

For the feature of claim 40 see claim 40 rejection. For rest of the features see claim 5 rejection.

8. Claims 10-13, 28-31, and 46-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 5, 836,014 by Robert Neil Faiman Jr. (hereinafter "Faiman"), in view of U.S. Patent No. 6,804,686 by Blake W. Stone et al. (hereinafter "Stone"), further in view of 'Introducing Microsoft DotNet', 07/02, 2002, by Christophe Lauer (hereinafter "Lauer").

CLAIM	Faiman / Stone / Lauer
10. The system of claim 1, wherein: a language interface of the plurality of language interfaces allows a first language module of the plurality of	For the feature of claim 1 see claim 1 rejection. Faiman and Stone teach all aspects of claim 10, but he does not mention 'first language module to

language modules to interact with a second language module of the plurality of language modules.

mention 'first language module to interact with a second language module' specifically, however, Lauer teaches it in an analogous prior art. In Lauer, page 4, "**.NET is multi-language**
With the .NET platform, Microsoft will **provide several languages** and the associated compilers, such as C++, JScript, VB.NET (alias VB 7) and C#, a new language which emerged with .NET.", further in page 5, 4th paragraph, "The fact that all the .NET languages are compiled in the form of an intermediate code also means that a class written in a language may be **derived in another language**, and **it is possible to instantiate in one language an object of a class written in another language.**" (*interact one language with another language, allowing first language to request a portion of the second language module*).

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman and Stone's disclosure of the multi-programming-language by the .NET taught by Lauer, for the purpose of providing the same services form in different languages (see Lauer, page 5, 7th paragraph) to improve compatibility via a common denominator.

11. The system of claim 10, wherein:
the language interface is operable to nested languages, allowing the first language module to request the

Same as claim 10 rejection.

language module to request the compilation of a specified portion of the computer program using the second language module.

12. The system of claim 1, wherein: at least one language module of the plurality of language modules is for Java language.

For the feature of claim 1 see claim 1 rejection. In Lauer, page 5, 9th paragraph, "The most symptomatic example concerns **Java. It is one of the intended .NET languages**, thanks to Rational, who are currently working on a **Java** to MSIL compiler."
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman and Stone's disclosure of using Java by the .NET taught by Lauer, for the purpose of providing the same services form different languages (see Lauer, page 5, 7th paragraph).

13. The system of claim 1, wherein: a second language module of the plurality of language modules extends a first language module of the plurality of language modules to provide a new language which is an extended version of a programming language compiled by the first language module.

For the feature of claim 1 see claim 1 rejection. Again, in Lauer page 5, 4th paragraph, "The fact that all the .NET languages are compiled in the form of an intermediate code also means that **a class written in a language may be derived in another language (*a new language*)**". Also in Lauer page 4, 6th paragraph, "All these languages are compiled via an **intermediate binary code**, which is independent of hardware and operating systems. This language is **MSIL: Microsoft Intermediate Language**. MSIL is then executed in the Common Language Runtime (CLR), which

	basically fulfills the same role as the JVM in the Java platform. MSIL is then translated into machine code by a Just in Time (JiT) compiler." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Faiman and Stone's disclosure of the multi-programming-language by the MSIL taught by Lauer, for the purpose of providing the same services form in different languages (see Lauer, page 5, 7th paragraph) to improve compatibility via a common denominator.
28. The method of claim 19, further comprising: allowing a first language module of the plurality of language modules to interact with a second language module of the plurality of language modules.	For the feature of claim 19 see claim 19 rejection. For rest of the features see claim 10 rejection.
29. The method of claim 28, wherein: the first language module is operable to request the compilation of a specified portion of the computer program using the second language module.	For the feature of claim 28 see claim 28 rejection. For rest of the features see claim 11 rejection.
30. The method of claim 19, wherein: at least one language module of the plurality of language modules is for Java language.	For the feature of claim 19 see claim 19 rejection. For rest of the features see claim 12 rejection.
31. The method of claim 19, further comprising: extending a first language module of	For the feature of claim 19 see claim 19 rejection. For rest of the features see claim 13 rejection.

the plurality of language modules using a second language module of the plurality of language modules to provide a new language which is an extended version of a programming language compiled by the first language module.

46. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to:
allow a first language module of the plurality of language modules to interact with a second language module of the plurality of language modules.

For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 10 rejection.

47. The machine readable medium of claim 37, wherein:
the first language module is operable to request the compilation of a specified portion of the computer program using the second language module.

For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 11 rejection.

48. The machine readable medium of claim 37, wherein:
at least one language module of the plurality of language modules is for Java language.

For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 12 rejection.

49. The machine readable medium of claim 37, further comprising instructions that when executed cause the system to:
extend a first language module of the plurality of language modules using a second language module of the plurality

For the feature of claim 37 see claim 37 rejection. For rest of the features see claim 13 rejection.

of language modules to provide a new language which is an extended version of a programming language compiled by the first language module.

Conclusion

The following summarizes the status of the claims:

35 USC § 102 claim rejection: 1, 7, 8, 19, 25-26, 37, 43-44, and 55-56.

35 USC § 103 claim rejection: 2-6, 9-18, 20-24, 27-36, 38-42, and 45-54.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2122

CC

TUAN DAM
SUPERVISORY PATENT EXAMINER